УДК 001:001.895

И. И. Сомов (somovlife@gmail.com),
магистрант
В. В. Бондарева (v_bond@rambler.ru),
канд. техн. наук, доцент
Белорусский торгово-экономический университет потребительской кооперации
г. Гомель, Республика Беларусь

АНАЛИЗ МЕТОДОЛОГИЙ И ИНСТРУМЕНТОВ КОНТРОЛЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (AGILE & SCRUM)

В данной статье рассматриваются методологии и инструменты контроля разработки программного обеспечения на примере Agile и Scrum.

This article explores the methodology and tool for monitoring software development on the example of AGILE and SCRUM.

Ключвые слова: методологии разработки ПО; Agile; Scrum.

Key words: software development methodologies; Agile; Scrum.

Разработка программного обеспечения (ПО) – это один из ключевых моментов, определяющих мир информационных технологий. Современные технологии развиваются настолько быстро, что на данный момент появилось такое определение, как *технологическая сингулярность* – по мнению некоторых исследователей, это короткий период чрезвычайно быстрого технологического прогресса. Все это возможно в случае развития компьютерных технологий и выпуска нового, усовершенствованного, инновационного программного обеспечения. В данной статье будут рассмотрены подходы к разработке ПО и некоторые инструменты, помогающие этот процесс контролировать.

Разработка ПО – это структура, согласно которой построен процесс создания ПО. Термин «структура» можно заменить на «видение» или «образ мышления». Чтобы понять, как организовать данный процесс, следует рассмотреть его отдельные шаги. На данном этапе происходит разветвление этого понятия на модели разработки ПО, которых большое количество. В данной статье для получения наиболее общей картины их количество будет сокращенно до трех доминирующих в этой области моделей:

1. Водопадная (каскадная) – модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.

2. Итерационная – это выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы. Проект при этом подходе в каждой фазе развития проходит повторяющийся цикл PDCA: Планирование – Реализация – Проверка – Оценка (plan-do-check-act cycle).

3. Спиральная – модель, отличительной особенностью которой является специальное внимание рискам, влияющим на организацию жизненного цикла. Каждый виток спирали соответствует созданию фрагмента или версии программного обеспечения и разбит на четыре сектора:

- определение целей;
- оценка и разрешение рисков;
- разработка и тестирование;
- планирование следующей итерации.

Каждая отдельная черта данных моделей проектирования имеет весьма значительный эффект в глобальном контексте. Отхождение от модели жизненного цикла разработки конкретного ПО могли бы привести к огромным финансовым и временным потерям. Именно для исключения подобных последствий были разработаны отдельные методологии. В процессе рассмотрения методологий происходит непосредственное соприкосновение с людьми, и на этом этапе рассмотрения методов проектирования станут наиболее очевидны причины их применения.

Существует несколько основных методологий, которые используются наиболее часто:

- Agile;
- Cleanroom Software Engineering;
- Rapid Application Development (RAD);
- Test-driven Development (TDD).

Cleanroom Software Engineering (методология «чистой комнаты») — процесс разработки программного обеспечения, предназначенный для создания программного обеспечения с сертифицируемым уровнем надежности.

RAD (rapid application development – быстрая разработка приложений) – концепция создания средств разработки программных продуктов, уделяющая особое внимание быстроте и удобству программирования, созданию технологического процесса, позволяющего программисту максимально быстро создавать компьютерные программы.

Test-driven Development (TDD) – техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и в конце проводится рефакторинг нового кода к соответствующим стандартам.

Некоторые из вышеперечисленных методологий достаточно часто и успешно совмещают, например Agile и TDD. Методология Agile является наиболее распространенной в современных компаниях по разработке программного обеспечения. Ее используют Yandex, EPAM, Wrike и др.

Гибкая методология разработки (Agile software development, agile-методы) – серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля.

Данная методология имеет свой манифест, который, в свою очередь, переведен на 68 языков. Его суть выражается в четырех основных ценностях:

- *люди и взаимодействие* важнее процессов и инструментов;
- *работающий продукт* важнее исчерпывающей документации;
- *сотрудничество с заказчиком* важнее согласования условий контракта;
- *готовность к изменениям* важнее следования первоначальному плану.

Таким образом, важность того, что справа, не отрицается, однако приоритет отдается ценностям, расположенным слева (помечены курсивом).

Следует отметить, что это не список правил, а образ мышления, сформированный на основе этих четырех ценностях:

1. Создавая программный продукт, нужно понимать, что его основная цель – взаимодействие людей с людьми посредством ПО, а не людей с продуктом.

2. Работая над созданием ПО, следует стремиться к созданию продукта, который будет удовлетворять требованиям заказчика, а не просто выполнять определенный список функции. Только исходя из такого рассуждения будет достигнута наибольшая эффективность работы команды разработчиков.

3. Дружеское отношение и вхождение в суть проблемы являются эффективными способами достижения наибольшей продуктивности, нежели строгое регулирование отношений посредством технического задания и других административных документов.

4. Готовность изменять код говорит о том, что структура ПО изначально строиться более гибко и модульно. Каждый модуль подлежит быстрой замене в случае надобности доработки или изменения начальной постановки задачи.

Эти четыре правила итеративной модели методологии Agile имеют значительную эффективность по сравнению с Waterfall-моделью, которая является строго регламентированной и значительно менее гибкой.

Методология Agile была взята за основу при создании ее частных реализаций (Scrum, XP, Lean, FDD). Наиболее часто встречающиеся из них – Scrum. Это методология гибкой разработки ПО, которая делает акцент на качественном контроле процесса разработки.

Методология Scrum на данный момент занимает лидерские позиции среди методологий ведения проектов. Это обусловлено рядом причин:

1. Бюджет проекта снижается за счет линейных затрат. В каскадной модели затраты располагаются на s-кривой из-за неравномерной занятости специалистов, а в циклической модели период разработки занимает более длинный отрезок времени, что также негативно влияет на бюджет проекта.

2. Гибкость и удобство для заказчика.

Чтобы понять, как функционирует Scrum, следует посмотреть на его реализацию изнутри.

Scrum – это набор принципов, на которых строится процесс разработки, позволяющий в жесткофиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определен наибольший приоритет.

Схематично Scrum можно разделить на 13 обязательных элементов (три роли, шесть митингов и четыри документа), которые, взаимодействуя друг с другом, образуют процесс разработки:

1. Product owner (PO – владелец продукта) – человек с видением, который может ответить на вопросы (часто это и есть вкладчик): что нужно делать? кому нужен данный продукт? где получить финансирование?

2. Product backlog (журнал пожеланий проекта) – условно, это документ, в котором PO описывает полностью проект, разбитый на отдельные модули. Это список пожеланий, подлежащих реализации.

3. Команда – группа людей, которая включает в себя всех технических специалистов нужных для реализации поставленных PO-задач.

4. Scrum master – командный куочер или человек, который отвечает за взаимодействие специалистов в команде.

5. Планирование спринта – встреча, на которой команда принимает решение о реализации определенного модуля проекта в установленный период (спринт – 1–4 недели).

6. Sprint backlog (журнал пожеланий спринта) – описание модуля, который будет реализован за предстоящий спринт.

7. Daily standup – встреча, которая проходит каждый день между участниками команды и осуществляется для синхронизации разработки.

8. Работа с журналом пожеланий – отдельная встреча, которая организуется в случае, если для реализации некоторых модулей нужны специфические знания и специалисты, т. е. для устранения данного пробела потребуется время. Это обсуждается заранее и вопрос решается на ранних стадиях.

9. Scrum of scrums – встреча представителей команд для синхронизации работы уже на уровне команд. Следует помнить, что каждая команда может работать над одним проектом, но над разными модулями, нередко данные модули имеют зависимость друг от друга. Эта встреча организуется для согласования данных зависимостей.

10. Demo – встреча, на которой происходит демонстрация продукта сделанного за спринт.

11. Ретроспектива – встреча, на которой происходит оценка самой работы команды (оценка качества взаимодействия и принятия решений).

12. Продукт (Potentially Shippable Product increment) – прирост в функционале реализуемого продукта.

13. Критерий готовности (Definition of Done (DoD)) – критерий, определяющий степень готовности элемента из журнала пожеланий пользователя.

Эти 13 элементов должны обязательно присутствовать, чтобы разработка была в рамках данной методологии. Они направлены на детальное отслеживание процесса разработки проекта. Таким образом, в течение спринта команда должна предоставить готовый результат, который можно «потрогать», и оценить прогресс и перспективы. Например, в каскадной модели между постановкой задачи и оценкой может пройти большой промежуток времени, сама оценка также может пройти негативно, вследствие чего будет потрачено много времени на доработку. При осуществлении контроля в короткие сроки разработка проекта становится гибкой (можно постоянно вносить изменения), а также исключается риск значительных потерь времени.

Несмотря на положительные стороны, Scrum имеет «узкие места»:

- не дает четких сроков на полную разработку проекта, поэтому его чаще можно не закончить, а прекратить;
- при удалении команд друг от друга процесс разработки может стать очень тяжелым в плане синхронизации работы команд;
- отсутствие четкого технического задания может превратить процесс разработки в множество повторных итераций для доработки и переработки продукта (можно рассматривать двояко);
- работа в Agile мотивирует разработчиков решать все поступившие задачи простейшим и быстрейшим способом, что приводит к снижению качества продукта и накоплению дефектов.

Scrum master обязан знать о всех рисках и «узких местах» методологии, создавая при этом условия, избегающие ее слабые стороны.

Кроме управления проектами по разработке ПО, Scrum может также использоваться в работе команд поддержки программного обеспечения или как подход к управлению разработкой и сопровождению программ.

Данная методология уже успела себя зарекомендовать и стать эталоном во многих организациях. На данный момент можно встретить заказчиков ПО, которые вовсе отказываются от сотрудничества с разработчиками, работающими не по методологии Scrum. И это во многом оправдано.

Подводя итоги, стоит отметить несколько моментов:

- принципы, которые применяются в методологии Scrum доступны к применению не только в IT-областях, но и, например, в образовательной деятельности;
- горизонтальный характер отношений между членами команд и постепенный отказ от вертикальной иерархии в коллективе набирают обороты и имеют положительный эффект.

Методология Scrum ориентирована на людей и их взаимодействие, что делает ее применение более широким и оправданным.

Список использованной литературы

1. **Manifesto** for Agile Software Development: манифест методологии Agile [Электронный ресурс]. – Режим доступа : http://agilemanifesto.org. – Дата доступа : 15.01.2017.

2. **Scrum**: Правила Игры – Habrahabr // Главный некоммерческий российский портал программистов [Электронный ресурс]. – Режим доступа : http://habrahabr.ru/company/wrike/blog/315580. – Дата доступа : 15.01.2017.

3. **Scrum** // Википедия [Электронный ресурс]. – Режим доступа : http://ru.wikipedia.org/wiki/Scrum. – Дата доступа : 15.01.2017.